Duration: **150 minutes**
Aids Allowed: ***None***

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|

UTORid: _____

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 8 questions on 16 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

MARKING GUIDE

1: _____/ 1
2: _____/ 7
3: _____/ 7
4: _____/ 5
5: _____/ 4
6: _____/ 7
7: _____/ 5
8: _____/ 6

TOTAL: _____/42

Good Luck!

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 1. [1 mark]

Draw a diagram of an empty tree

Question 2. [7 marks]

Part (a) [3 marks]

Let f and g be two functions. Give the formal definition of $f = O(g)$

Part (b) [2 marks]

On your answer to part a), underline the part(s) that convey the idea of "within a constant factor"

Part (c) [2 marks]

From our class discussion, explain why we needed to include the idea of "within a constant factor" in the definition of big-Oh

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 3. [7 marks]

Part (a) [4 marks]

What is the output of the following program?

```
def weird_version(s):
    """ (str) -> str
    Return a weird version of string s.
    """
    if len(s) < 2:
        return s
    else:
        return s[1] + weird_version(s[2:]) + s[0]

if __name__ == "__main__":
    print(weird_version("A48WEIRD"))
    print(weird_version("ABC12345DEF"))
```

Part (b) [3 marks]

Suppose we made a new version of the function weird called weirdplus, where we changed the line
`if len(s) < 2:`
to
`if len(s) < 3:`
What can we say about a string s if we know that weird(s) != weirdplus(s)?

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 4. [5 marks]

Consider the following class for a binary tree node.

```
class BTNode:
    def __init__(self, left=None, right=None):
        """ (BTNode, BTNode, BTNode) -> NoneType
        Initialize a new BTNode with pointers to subtrees left and right.
        """
        self.left = left
        self.right = right
```

Using the above class, write a function called `sum_depths`, which takes the root of a binary tree, and returns the sum of the depths of all the nodes in that tree. For example, if r is the root of a tree that has 1 node at depth 0, 2 at depth 1, 3 at depth 2, and 5 at depth 3, then `sum_depths(r)` should return 23 (because $1 * 0 + 2 * 1 + 3 * 2 + 5 * 3$ equals 23).

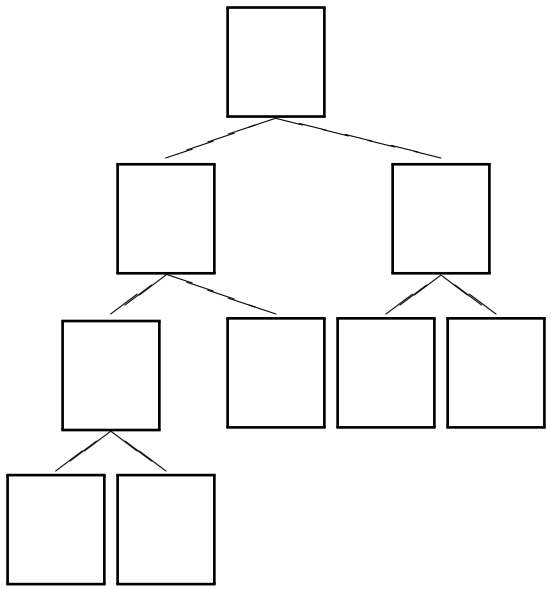
Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 5. [4 marks]

Part (a) [2 marks]

Suppose the numbers 1,2,3,4,5,6,7,8,9 were found in a min heap which is shaped like the following tree.

The numbers could have been inserted in any order.



Mark with an X all the nodes where it is possible to find the number 3.

Part (b) [2 marks]

Instead of using a tree, we can hold the same heap in the list below.

Mark with an X all the places in the list where it is possible to find the number 7.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 6. [7 marks]

The following classes were defined in the starter code of your regular expression assignment.
`UnaryTree, BarTree, BinaryTree, DotTree, Leaf, RegexTree, StarTree.`

Part (a) [3 marks]

For each of these classes, give its child classes. Write "none" if a class has no subclasses.
To help you get started, we completed the question for the first class.

- Children of UnaryTree: StarTree.
- Children of BarTree:
- Children of BinaryTree:
- Children of DotTree:
- Children of Leaf:
- Children of RegexTree:
- Children of StarTree:

Part (b) [4 marks]

StarTree is the only subclass of UnaryTree. So by defining StarTree as a child class of the parent of UnaryTree, we could have avoided the need for the UnaryTree class altogether. Give two reasons why our design included the UnaryTree class.

Reason #1:

Reason #2:

```
def del_index(head, tail, index2del):
    """ (DLLNode, DLLNode, int) -> (DLLNode, DLLNode)

    Remove the item at index index2del from the doubly linked list with
    head head and tail tail (both None if list empty).
    Return the head and tail of the updated list.

    If the items were stored in a python list L in the same order, then the
    effect of executing del_index(head, tail, index2del) should be the same
    as that of executing del L[index2del].
    Note how index2del can be negative.

    REQ: -N <= index2del < N, where N is the number of items in the list.
    """

# find the node to delete
# if deleting head
# if deleting tail
# start at back of list
# start at front of list
# unlink node to delete
curr = curr.next_link
curr = curr.prev_link
curr = head
curr = tail
curr.next_link.prev_link = curr.prev_link
curr.prev_link.next_link = curr.next_link
else:
else:
else:
head = curr.next_link
if (curr.next_link == None):
if (curr.prev_link == None):
if index2del >= 0:
index2del += 1
index2del -= 1
return (head, tail)
tail = curr.prev_link
while (index2del < -1):
while (index2del > 0):
```

Question 7. [5 marks]

For the doubly linked list exercise, Nick had written a well documented and correct helper function. Then the CODE MANGLER struck. Dun dun dunnn ... He (she?) did the following to all of Nick's code after the docstring, including comments.

- Deleted all blank lines.
- Removed all leading spaces (i.e., removed all indentation).
- Permuted, or shuffled, all the lines.

As a result, Nick's code wound up as it appears on the opposite page.
In the space below, reconstruct Nick's original code (you don't need to re-write the header or docstring).

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Question 8. [6 marks]

Part (a) [2 marks]

Draw the binary search tree (BST) we would get if we were to insert the letters M A N G L E R in the order listed (i.e., M is the first letter added to the tree).
In case you need it, here are all the upper case letters in alphabetical order.
`A B C D E F G H I J K L M N O P Q R S T U V W X Y Z`
Advice: Do this part very carefully. Any error here will likely lead to wrong answers in part (b) also.

Part (b) [4 marks]

Perform two rotations on the BST from part (a) so to have G at the root. Draw the tree after each rotation.

On this page, please write nothing except your name.

Last (Family) Name(s): _____

First (Given) Name(s): _____

Total Marks = 42